HAL
archives-ouvertes.fr

Application of a Co-evolutionary Genetic Algorithm to solve the Periodic Railway Timetabling Problem

Diego Arenas, Rémy Chevrier, Clarisse Dhaenens, Said Hanafi, Joaquin Rodriguez

▶ **To cite this version:**

Diego Arenas, Rémy Chevrier, Clarisse Dhaenens, Said Hanafi, Joaquin Rodriguez. Application of a Co-evolutionary Genetic Algorithm to solve the Periodic Railway Timetabling Problem. IESM 2013, 5th international conference on industrial engineering and system management, Oct 2013, Rabat, Morocco. 10p. hal-00909588

HAL Id: hal-00909588
https://hal.archives-ouvertes.fr/hal-00909588

Submitted on 26 Nov 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

International Conference
on Industrial Engineering and Systems Management
IESM'2013
October 28 - October 30
RABAT - MOROCCO

Application of a Co-evolutionary Genetic Algorithm to solve the Periodic Railway Timetabling Problem ⋆

Diego ARENAS [a], Rémy CHEVRIER [a], Clarisse DHAENENS [b], Said HANAFI [c], Joaquin RODRIGUEZ [a]

[a] *Univ. Lille Nord-de-France, IFSTTAR - 20, rue Élisée Reclus, 59666 Villeneuve d'Ascq Cedex, France*

[b] *INRIA - Parc Scientifique de la Haute Borne, 40, Avenue Halley, Bât A, Park Plaza, 59650 Villeneuve d'Ascq, France*

[c] *UVHC - Campus Mont Houy, 59313 Valenciennes Cedex, France*

Abstract

In train operations, a timetable is used to establish the departure and arrival times for the trains at the stations or other relevant locations in the rail network or a subset of this network. The elaboration of a timetable responds to the commercial needs of the customers, for both passenger and freight traffic, but also, it must respect some security and capacity constraints. The combination of these requirements and constraints makes the preparation of a yearly timetable a complex process that usually takes months to be fully completed. This paper addresses the problem of generating periodic timetables, which means that the trains concerned are operated on a recurrent pattern, e.g., trains of the same line will run every 30 minutes, we present a suitable constraint-based model of the problem. Furthermore, we propose a dedicated genetic algorithm, based on a co-evolutionary scheme with two populations, to create feasible and quality periodic timetables in short periods of time. Finally, two case studies are discussed, both of them representing a subset of the Netherlands railway network.

Key words: Genetic Algorithms, PESP, Railway, Scheduling, Timetabling

1 Introduction

The train timetabling problem involves mainly two procedures: First, to define the departure and arrival times for the trains at the stations or other relevant locations in the rail network or a subset of this network. And second, to establish the platforms and route plans, i.e., the assignment of each train to an appropriate platform and corresponding inbound and outbound routes in every station. For a timetable to be viable or feasible, it must respect some constraints that may be hard constraints: related to safety and capacity issues, e.g., two trains cannot be in the same track segment at the same time, or soft constraints: related to specific requirements for the timetable, e.g, to establish particular passenger connections between two lines of trains.

It is desired that the generation of the arrival and departure times and the selection of the routes through the stations are carried out simultaneously. However, real-size problems are too large and have thousands of

⋆ This paper was not presented at any other revue. Corresponding author D. Arenas. Tel. + 33 (0)3 20 43 85 06. Fax : + 33 (0)3 20 43 83 59.

Email addresses: diego.arenas@ifsttar.fr (Diego ARENAS), remy.chevrier@ifsttar.fr (Rémy CHEVRIER), clarisse.dhaenens@lifl.fr (Clarisse DHAENENS), said.hanafi@univ-valenciennes.fr (Said HANAFI), joaquin.rodriguez@ifsttar.fr (Joaquin RODRIGUEZ).

constraints. Even modern approaches, using the current available technology are not able to produce a result within an acceptable amount of time. Therefore the two parts of the problem are treated separately: first the timetable is generated and only when a feasible timetable is found, the routings through the stations are determined. However, at this point a feasible routing plan for the given timetable may not be possible, then it is necessary to come back to the previous step and generate another viable train timetable. Many iterations may be necessary in order to finally produce both a viable timetable and routing plan.

In this paper, we focus on the generation of a feasible train timetable, therefore the platforming and routing plans are not discussed. Additionally, besides the timetabling problem, there are two other procedures that are closely related to the railway timetable design problem, see Figure 1. First: the rolling stock plan: which specifies among other things, which engines and carriages will be used for each train line. And second, the crew schedule, which defines the working plan for each person involved on train trips, such as drivers, controllers, etc.

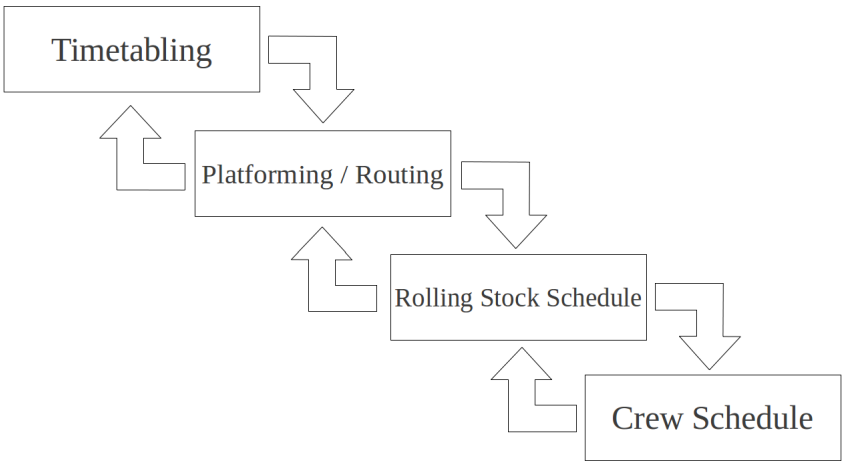

Fig. 1. Timetable Planning Problem

On each of the steps described above, an optimization process can be carried out, which consists, for example, in making minor modifications to a feasible timetable in order to optimize it, e.g., to reduce the total number of trains, or minimize the overall waiting times for passengers during connections, or reduce the number of platforms needed in some station at some given time.

Traditionally, the timetable planning problems were solved based on the experience and expertise of the involved railway planners. But at the early nineties it was recognized within the railway community that the application of models and techniques for supporting the solution process of these problems may be truly advantageous. It would not only lead to better solutions, but it also conducts to a considerable reduction of the time needed for the planning process.

There are mainly two types of timetables, periodic and non-periodic. The difference between them is that: Non-periodic timetables are designed based only on the set of requests for trips using the network, i.e., try to allocate the departure and arrival times corresponding to the requested departure and arrival times of the trip. While periodic timetables are built based on a time period T, a timetable pattern is created for this period where all trips are scheduled. Then this pattern is repeated to cover the whole day. This means that each line of passenger trains is operated in a regular recurrent way, e.g., the trains of the same line run every 30, 60 or 120 minutes. See Table 1 for an example of a periodic timetable.

Table 1
Periodic timetable for two TGV (High-Speed Train) lines connecting the cities of Paris (Paris-Nord train station) and Lille (Lille-Flandres and Lille-Europe train stations) [**?**]

Line	Direction	Departure.	Arrival	Direction	Departure	Arrival
Paris-Nord $->$ Lille-Flandres	Lille	8:46	9:48	Paris	10:11	11:14
Paris-Nord $->$ Lille Europe	Lille	9:46	10:45	Paris	11:13	12:15
Paris-Nord $->$ Lille-Flandres	Lille	10:46	11:48	Paris	11:41	12:44
Paris-Nord $->$ Lille Europe	Lille	11:46	12:45	Paris	13:13	14:14
Paris-Nord $->$ Lille-Flandres	Lille	12:46	13:48	Paris	15:11	16:14

One advantage of a periodic timetable is that passengers can easily keep in mind the departure time of their train at their station. However, a drawback is that these kinds of timetables are not very flexible, and it is difficult to offer a large number of direct connections. Another disadvantage, is the fact that a completely periodic timetable may be rather inefficient: trains might have to be operated even at some specific times of a day with only a small number of passengers. Therefore, in practice there are usually exceptions to the completely periodic timetable. For example, there may be some additional trains during rush hours, and frequencies may be reduced during late evening hours [2]. In this paper we discuss only the generation of periodic timetables.

Train stations may have many platforms and complex track layouts, which could also be used for resolving timetable conflicts and soft constraints. In this work, we assume that each station has enough number of platforms to accommodate the trains that will finally stop on it.

Finally, in this paper, a line plan is assumed to be known a priori, i.e., the list of lines, and the stop schedule for each line, as well as the desired connections between lines.

The rest of this paper follows the next structure: Section 2 reviews some popular approaches that address this problem. Section 3 presents the formulation used in our approach. Section 4 describes how we use genetic algorithms to solve the train periodic timetable generation problem. Section 5 presents the case studies and discusses the obtained results using our technique.Finally, in Section 6, we conclude this paper and present some perspectives for further works.

2 Brief Literature Review

There are numerous authors that developed different techniques to address scheduling problems and specifically the train timetabling problem, in this section we will briefly discuss some of the most relevant works that deal with this topic.

Serafini *et al.* [**?**] introduce the Periodic Event Scheduling Problem (PESP), which is a framework for scheduling periodic activities. The PESP is proven to be an exceptionally rich model that can be used in many applications, e.g., airline scheduling, traffic light scheduling and of course, train scheduling, among others.

Odijk [6] proposes the PCG (PESP Cut Generation) algorithm to generate timetables for train operations using a mathematical model consisting of periodic time window constraints as input. His model is based on the definition of constraints applied on a mathematical formulation to solve the problem.

Kroon *et al.* [2] separate the problem in two parts: the generation of the departure/arrival times and the selection of the routes through the stations. The first part describes a mathematical formulation based on the PESP to solve the scheduling problem, and since the second part is out of scope of this paper it will not be described. On the first part, he defines a set of different types of constraints and then use them as input to the PESP solver, obtaining then a timetable that will be used as input to the second part.

Liebchen's [**?**] work is also based on the PESP, however two natural optimization variants are presented and discussed: The Max-T-Pesp, and a heuristic method called Cut-Heuristic. Then these two techniques are applied to solve an optimization problem on a real scenario: the subway system of Berlin. The cut-heuristic algorithm is based on two optimization-based observations for the PESP.

Caprara *et al.* [1] concentrate on the problem of a single, one-way track linking two major stations, with a number of intermediate stations in between. They use a directed multi-graph in which nodes correspond to departures and arrivals at a certain station at a given time instant. Then they use this formulation to derive an integer linear programming model which is relaxed in a Lagrangian way. The objective is to maximize the sum of the "profits" of the scheduled trains, the profit achieved for each train depends on: the "shift" defined as the absolute difference between the departure times from a given station in the ideal and actual timetables; and on the "stretch" defined as the non-negative difference between the running times in the actual and ideal timetables. Although Caprara *et al.* model presented in this work is not applied to create a periodic timetable, it is highly flexible and it can be extended to the PESP model and then used to generate periodic timetables.

Up to this point, the authors base their approaches on the resolution of complex linear programming models. Other authors use evolutionary techniques such as genetic algorithms to completely or partially address the

problem. As discussed in [**?**], Genetic Algorithms are proven to be a solid method to deal with the periodic timetabling problem, and their behavior is more stable than other techniques when used both in small and big instances of the problem.

As an instance of partial use of genetic algorithms, Semet's *et al.* [7] approach focus on the reconstruction of a timetable following a small perturbation. They try to minimize the total accumulated delay by adapting the departure and arrival times for each train and allocation of resources. They use a permutation-based evolutionary algorithm to gradually reconstruct the schedule. The railway network can be seen as a graph where nodes are stations or switches and where interconnecting edges eventually hold several tracks for trains to use. Their objective function consists in minimizing the total accumulated delay (for all trains at all nodes). The algorithm is a hybrid, or memetic algorithm, in the sense that it combines an evolutionary engine with a mathematical programming tool, called ILOG CPLEX. The evolutionary part of the algorithm is used to quickly obtain a good but suboptimal solution, which is fed to CPLEX as an initial solution which will search for the global optimum.

An example of complete use of genetic algorithms to address the problem, Kwan and Mistry [3] use a co-evolutionary algorithm for the automatic generation train timetables, essentially at the early stages of the process, trying to allocate as many capacity requests as possible, and discover the conflicts that have to be solved by the train operating companies. They assign some penalty weights to each type of constraint violation thus the Objective function is to Minimize the weighted sum of violations. They use three types of populations which are evolved one after another: "departing times", "running times" and "resources options". The combination of three individuals i.e one of each population, result on a timetable. On each step of the evolution a timetable is generated and evaluated, once the termination condition is achieved, the algorithm is stopped and the best timetable is given as result.

3 Problem Formulation

In this section we describe the model for our solution, first we shall define the data, the variables and the objective function. And then, we describe the constraints that our model takes into account.

Data:
T : Period for one cycle.
i, j : Trains.
s, t : Stations.
L, U : Lower and upper time limits for some constraints.

Variables:
$D_{i,s}$: Departure time of train i from station s
$A_{i,s}$: Arrival time of train i to station s
$Q_{e,e'}$: Binary value. If the events e and e' take place on the same period T the value of $Q_{e,e'}$ is 0, or 1 otherwise.

Objective Function: The objective function is defined as the minimization of the weighted sum of constraint violations. The weight for each type of constraint violation depends on the importance of the constraint, e.g., violating a headway constraint is a lot more serious than violating a connection constraint.:

$$Minimize[(W_r \times V_r) + (W_h \times V_h) + (W_s \times V_s) + (W_c \times V_c)]$$

$$Minimize \sum (W_x \times V_x) \tag{1}$$

Where:
W_x : Is the weight value corresponding to a violation of a constraint type x.
V_x : Represents the constraint violation count of type x
x: Type of constraint, can acquire one of the following values: r,h,s,c. Corresponding to: Running-time, Headway, Single-track and Connection.

Constraints:
Running Time: Basic constraint, it will ensure that the travel time for some train i departing from station

s with destination station t, is between the specified limits Le and Ue previously calculated. (Lower Elapsed Time and Upper Elapsed Time, respectively).

$$Le_{i,s,t} \leq A_{i,t} - D_{i,s} + T \times Q_{A_{i,t},D_{i,s}} \leq Ue_{i,s,t} \tag{2}$$

Where: $Le_{i,s,t}$ and $Ue_{i,s,t}$ are determined based on the details of the infrastructure between the stations s and t (including the safety system) and the running time characteristics of the involved rolling stock.

Dwell Time: This constraint will guarantee that the stopping time of train i at station s is between the Lr and Ur. (Lower Rest Time and Upper Rest Time, respectively).

$$Lr_{i,s} \leq D_{i,s} - A_{i,s} + T \times Q_{D_{i,s},A_{i,s}} \leq Ur_{i,s} \tag{3}$$

Where: $Lr_{i,s}$ and $Ur_{i,s}$ are predefined values guaranteeing that there is enough time for passenger to board and disembark the train i. Also, if the train i have a connection at station s, this dwelling time should consider it too.

Connections: This kind of constraint will permit the passengers connections between trains, the time between the arrival of train i and the departure of train j must be within the limits of Lc and Uc (Lower Connection Time and Upper connection time, respectively).

$$Lc_{i,j,s} \leq D_{j,s} - A_{i,s} + T \times Q_{D_{j,s},A_{i,s}} \leq Uc_{i,j,s} \tag{4}$$

Where: $Lc_{i,j,s}$ and $Uc_{i,j,s}$ are defined as the estimated time needed for a passenger from train i to disembark, find the connection platform and board the train j within station s; larger stations normally require longer connections times.

Headway: Two trains i,j departing (arriving) from (to) station s, must have a headway time between them limited by Lh and Uh (Lower Headway Time and Upper Headway Time, respectively)

$$Lh_{i,j,s} \leq D_{j,s} - D_{i,s} + T \times Q_{D_{j,s},D_{i,s}} \leq Uh_{i,j,s} \tag{5}$$

Where: $Lh_{i,j,s}$ and $Uh_{i,j,s}$ depends on the kind of trains: Inter-City (IC) or local (stop-train), and their destination, for example: if two trains i,j departing from station s, having the same final destination, but train i stops at every station and train j only at major stations, train i cannot be overtaken by train j on the track, then the lower bound $Lh_{i,j,s}$ is the minimum difference in running time from station s to the next available station t ($Le_{i,s,t} - Ue_{j,s,t}$) plus the minimum headway time (predefined for train i). The upper bound $Uh_{i,j,s}$ is the difference between the cycle T and the headway time for train j.

Headway - Single Track Opposite Trains: Similar to the Headway constraint, this one is used when there is only one track connecting two stations and there is a train i which goes and comes back, in that case the returning train is denoted by i'. The time between the departure of train i and the arrival of train i' from/to station s must be within limits Lh and Uh.

$$Lh_{i,i',s} \leq D_{i,s} - A_{i',s} + T \times Q_{D_{i,s},A_{i',s}} \leq Uh_{i,i',s} \tag{6}$$

Where: $Lh_{i,i',s}$ is two times the minimum running time from station s to the next available station t ($2 \times Le_{i,s,t}$) plus the minimum headway for the train i, and the value of $Uh_{i,i',s}$ is the difference between cycle time T and the minimum headway for the train i.

4 A co-evolutionary-based solving method

Our proposal to address this problem is to develop a co-evolutionary genetic algorithm inspired by Kwan *et al.* [3] and Semet *et al.* [7] approaches. A comprehensive description about genetic algorithms can be found at [**?**] and [**?**]. Our model takes into account the objective function (1) and constraints (2) to (6) described in Section 3 and it supports the generation of periodic train timetables. There are two main parts which are detailed below, an illustrative schema is shown in Figure 2.:

(1) Data representation: In order to represent a timetable we are using two vectors:

- The first vector is a set of integers that represents the departure time of each train line at each station, e.g., if the instance of the problem have 3 lines and each line have 2 intermediate stop stations, the size of this vector would be 9: the first 3 integers are the departure times from the first station, the departure time from the first intermediate station and the departure time from the second intermediate station, the same is for the other 2 train lines.
- The second vector is a set of binary values. Each value represents whether the lower or higher running/dwelling time is currently selected, for all journeys and stops of each line. A "0" indicates that the lower (fastest) time is selected, and "1" represents the higher (slowest). These two values are defined a priori. For example the vector [011] represents that the train will make the first journey at its nominal speed (fastest), then it will dwell at the first intermediate station for a "longer" period of time, and will make the second journey at its slower allowed speed.

(2) The co-evolutionary Algorithm: The main algorithm is described in Algorithm 1. The essential components of the algorithm are briefly described next:

- Population: The algorithm works with two populations: DT: Departure times and RT: Running/Dwelling times. The individuals of each population are represented by the vectors described above. By combining one individual of each population, a timetable can be built.
- Variation Operator: The main operator we use at the evolution stage is mutation for both populations, additional operators can be easily added if necessary.
- Individual Selection: Used to select an individual from the populations in order to build a timetable, two selection criteria were implemented:
  - Discrete: A random individual is picked from the population.
  - By Tournament: Three individuals are randomly picked, the one with better fitness value is selected.
- Fitness Function: In order to evaluate one individual from one population, another individual from the other population is selected using one of the previously described selection methods. With these two individuals a new timetable is generated, then it is evaluated using the objective function described in Section 3, therefore, the fitness function consists in minimizing the total weight of the timetable. Additionally we added a small optimization criteria: we count the total number of physical trains that requires the generated timetable, more trains means heavier weight.

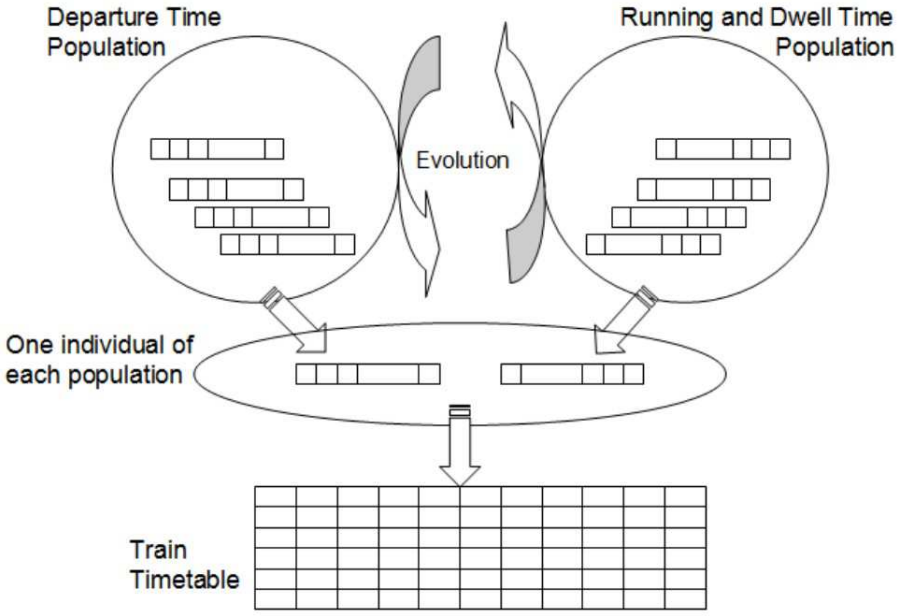

Fig. 2. Our approach design schema

5 Experimental Analysis

For experimenting and validating our work, we use two case studies, next a brief description of each one, they where extracted from [2] and [4] respectively. Then the results are presented.

The specifications of the computer used for both case studies are: Processor Intel Core i7 @2.9 GHz x4, 3.5 GB of RAM memory and using Ubuntu-64 v12.10 as operational system.

Algorithm 1 Solving the Train Timetable Generation Problem

```
Generate a random DT population                    ▷ DT stands for Departure Time
Generate a random RT population                    ▷ RT stands for Running and/or Dwelling Time
while Termination conditions not met do
    Evolve DT population using the variation operator.
    for each individual of DT do
        Select an individual from RT population using an individual selection technique.
        Combine the two individuals to generate a timetable.
        Evaluate the timetable using the fitness function.
    end for
    Evolve RT population using the variation operator.
    for each individual of RT do
        Select an individual from DT population using an individual selection technique.
        Combine the two individuals to generate a timetable.
        Evaluate the timetable using the fitness function.
    end for
end while
```

5.1 Description of the Instances

5.1.1 Case Study 1

Considers the triangle Amersfoort (Amf) - Deventer (Dv) - Zwolle (Zl) in the Netherlands, as depicted in Figure 3. The other relevant stations are Apeldoorn (Apd), Harderwijk (Hd) and Olst (Ost).

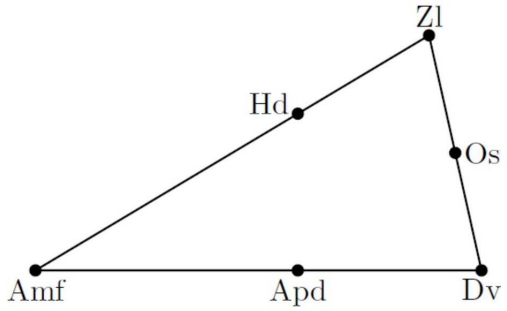

Fig. 3. Case Study 1: The triangle Amf - Dv - Zl [2]

In Table 2, the details of the line plan and the minimum running times are reported. Note that in practice the minimum running times may depend on the applied rolling stock type.

Table 2
Case Study 1 Line Plan [2]

Type	Route	Relevant stops	Min. runnning time	Freq.	Train
IC	Amf - Dv	Apd	24 + 10	1	1a,1b
IC	Amf - Zl	-	35	2	2a,2b,3a,3b
IC	Zl - Dv	Ost	14 + 6	1	4a,4b
stop	Amf - Zl	Hd	28 + 26	1	5a,5b

Here "IC" stands for Intercity and "stop" stands for local train or stop-train. The columns in the table should be read as follows: the IC Amf - Dv dwells in Apd with minimum running times of 24 minutes (Amf - Apd) and 10 minutes (Apd - Dv), respectively. It runs once every half an hour, that represents our cycle time. In the remainder of this section, we denote this train with number 1a in the direction of Dv and 1b in the opposite direction. Stations that are not mentioned in Table 2 are not relevant. That is why for the stop-train Amf - Zl, only the stop in Hd is listed in the table. The dwell times for the other stops are included in the minimum running times. The IC Amf - Zl does not stop in between Amf and Zl. However, to construct the timetable one should know the time it passes Hd. The minimum running times from Amf and Zl to Hd are 15 and 20 minutes, respectively.

5.1.2 Case Study 2

In Figure 4 a network with four lines is depicted; it is part of the Netherlands railway network, it is in fact an extension of the previous case study. However, and for sake of simplicity we are not detailing in this paper the input values for this instance, should the reader is interested, refer to [4].

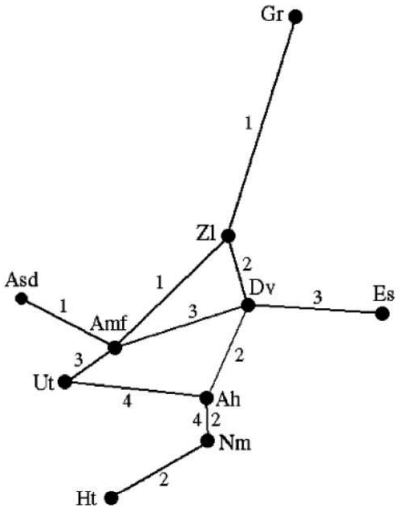

Fig. 4. Case Study 2: Subset of the Netherlands railway network [4]

5.2 Results

5.2.1 Case Study 1

Designing genetic algorithms requires to set some parameters, one of them is the size of the population which represents the total number of individuals or potential solutions. Other important parameter is the stop criteria, essentially, it establishes when the algorithm should stop and return a solution (even if it is sub-optimal, i.e., some constraints are violated).

Experimentation: In order to analyze and compare the impact of the population size in the resulting timetable, we perform 50 executions with a constant number of evaluations using the following permutation of parameters:

Total number of evaluations: (1 000 , 5 000 , 15 000) with a population of (40 , 80 , 160) individuals.

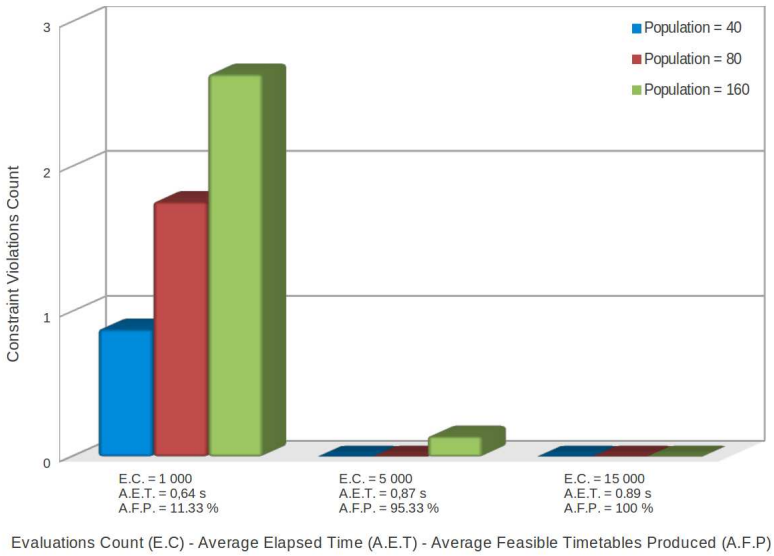

Fig. 5. Resuts of experimentation on Case Study 1

The results are displayed in Figure 5, as we can see, our solution can find a feasible train timetable (zero constraints violated) in 0.64 seconds (by this time, at least 11.33% of the generated timetables are feasible). After 0.87 seconds the probability to obtain a feasible timetable is 95.33%. Finally, after 0.89 seconds (or 15 000 evaluations), we can be sure that all timetables generated (100%) are feasible. By comparing the results with different size of populations, we can see that by increasing the size of the population we are actually reducing the number of generations that are evaluated (because of course each generation would need more evaluations) and thus, reducing the performance of the algorithm. We can conclude that for this sized instances of the problem, the number of generations offers greater improvements than the size of the population.

5.2.2 Case Study 2

As well as in the previous example, we analyze and compare the impact of the population size in the resulting timetable, we perform 40 executions with a constant number of evaluations using the following permutation of parameters:

Total number of evaluations: (0.8 M, 3 M, 5 M) with a population of (700 , 800 , 900) individuals.

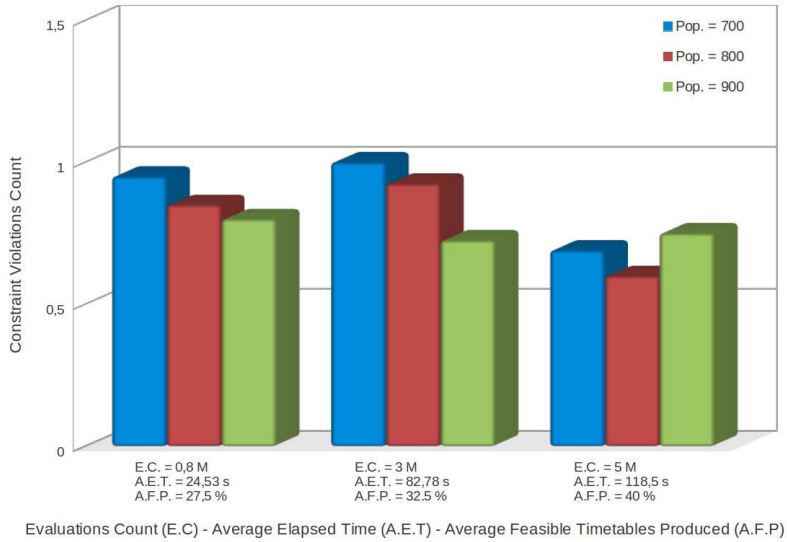

Fig. 6. Resuts of experimentation on Case Study 2

The results are shown in Figure 6, as we can see, our solution is able to generate a feasible timetable, for this instance of the problem, in less than 25 seconds. By this time, 27.5% of the generated timetables do not violate any constraint. Additionally, in average less than one constraint is violated, and only soft constraints (connections) are involved. After 118.5 seconds of execution, 40% of the generated timetables are feasible. In this case study, and unlike the precedent, the population size seems to affect the final solution in a different way. This can be explained because the size of this instance is larger than the precedent, a greater population offers a richer diversity of the potential solutions increasing the effectiveness of the genetic algorithm. Finally, during our experimentations we did not arrive to achieve a 100% of generated feasible timetables within the max evaluation count of 5 M. We believe that this was because the algorithm converged to local optimal solutions, and the value of the mutation operator was not sufficient to solve the problem.

6 Conclusion and Perspectives

The periodic train timetable design represents a complex but very interesting and exciting research subject. There are many different ways to address, model and solve the problem and the results of these may improve the current train timetable design system om the countries exploiting periodic timetables.

The results of our implementation were rather convenient for solving the periodic train timetable generation problem. Indeed, even though it represents a hard and complex problem, the performance of the genetic algorithm solving it, is remarkable. Our solution not only can rapidly find a feasible timetable for the problem,

but it also offers a flexible platform in which we can easily add, remove or modify the constraints in order to satisfy some other criteria. For example, it can be easily turned in a solution that focuses on the reduction of the overall waiting time for passengers. Also, by modifying the weight values of the constraints violations we can focus in the improvement of a particular type of constraints, for example by optimizing the connections between lines in the rail network.

However, our current solution does not deal with a very important issue in train timetable design, which is the planning of resources such as the track layout inside the stations, the number of platforms or the available rolling stock. Thus, further improvements to our approach should extend our model allowing it to handle this kind of resources in order to offer a more realistic solution.

Furthermore, in the experimentation process of our solution we have only tested the genetic algorithm with one kind of operator to evolve both populations: mutation; it would be necessary to test it with other kinds of operators in order to determine which one has better results with our data representation. This also concerns the individual selection process: tournament, as there are other techniques that may lead to a faster converging process of the algorithm. Fortunately, our flexible implementation allows a straightforward incorporation of those.

Additionally, we could improve our implementation according to a multi-objective paradigm, that is, deeply studying and analyzing how some constraints affect the final timetable. We could produce different kinds of options that would specialize in some particular interest, like optimization or finding a better connection plan to ensure the desired connectivity on the network. We believe that it can be done by simply adding new kinds of constraints or modifying the existing ones.

Finally, we can also combine our genetic algorithm with other optimization technique, such as mathematical programming, to create a hybrid solution. This can be made, not only to overcome the limitations of the genetic algorithms, e.g., a tendency to converge towards local optimal solutions, but also to take advantage of the strengths of both methods.

References

[1] A. Caprara, M. Fischetti, and P. Toth. Modeling and solving the train timetabling problem. *Operations Research*, 50(5):851–861, 2002.

[2] L.G. Kroon, D. Huisman, and G. Maroti. Railway timetabling from an operations research. Econometric Institute Report EI 2007-22, Erasmus University Rotterdam, Econometric Institute, June 2007.

[3] R.S.K. Kwan and P. Mistry. A co-evolutionary algorithm for train timetabling. Technical report, University of Leeds, July 2003.

[4] L. E. Meester and S. Muns. Stochastic delay propagation in railway networks and phase-type distributions. *Transportation Research Part B: Methodological*, 41(2):218 – 230, 2007. ¡ce:title¿Advanced Modelling of Train Operations in Stations and Networks¡/ce:title¿.

[5] K. Nachtigall and S. Voget. Minimizing waiting times in integrated fixed interval timetables by upgrading railway tracks. *European Journal of Operational Research*, 103(3):610–627, December 1997.

[6] M. A. Odijk. A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research Part B: Methodological*, 30(6):455 – 464, 1996.

[7] Y. Semet and M. Schoenauer. An efficient memetic, permutation-based evolutionary algorithm for real-world train timetabling. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 3, pages 2752–2759, 2005.